
# COMPARISON OF DIFFERENT FILE SHARING METHODS

Miguel Jiménez García

BSc Final thesis Project

Neptun ID: YQB5MS

Advisor Project: CZIRKOS Zoltán

Budapest University of Technology and Economics (BME)

## Index

## Introduction

The project is about P2P technology. Initially I will try to analyze the structures to carry out this type of application and the different ways in which each company has tried to maximize performance.

Also analyze the different versions that have appeared in the market, we will analyze in more detail the most used and also the different structures of networks that can perform data exchange between nodes.

Later, we will see how is the impact that P2P technology has had on society, with statistics of their use by users.

Finally, I have developed an application that sends information from one point to another between two connected nodes, in this case it is only text but could resemble P2P technology in which the shipment may be any type of file. We will see some examples of using the application and try to explain how I developed the application itself, looking for it the most important parts of the code developed.

## What is P2P?

P2P means Peer to Peer, not a network or software, but rather is defined as a network structure or a form of logical organization.

The principal use or what most it is used for share files between multiple users, as it has many more utilities but this is principally known. Later I will review other applications dealing P2P technology.

Is a computer network in which clients and servers change all time, nodes (Peers = each systems connected to network) behave as equal, contrary to client-server networks. That is, simultaneously act as clients and servers to other nodes on the network, that is, consumers and vendors can simultaneously.

P2P networks allow the direct exchange of information, in any format, between networked computers.

These networks do not use third parties (cache servers, origin servers ...), the file is transferred directly.

This means that P2P does not define a specific protocol or rules for use, P2P only indicates how they should be making connections and organizing nodes.

In P2P transfers can be very fast, because they have connections with all equipment capable of providing the service. Various parts are unloaded from

many different sources at the same time (Discharge Multipart).

The protocol is more complex than a client-server structure.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | BitTorrent | 31.69% | HTTP | 19.46% | BitTorrent | 20.32% |
| 2 | eDonkey | 18.23% | YouTube | 18.00% | HTTP | 17.70% |
| 3 | HTTP | 11.29% | BitTorrent | 17.20% | YouTube | 15.25% |
| 4 | YouTube | 5.24% | eDonkey | 6.96% | eDonkey | 9.39% |
| 5 | Skype | 2.48% | Flash Video | 5.62% | Flash Video | 4.70% |
| 6 | SSL | 2.47% | RTMP | 2.80% | RTMP | 2.47% |
| 7 | Teredo | 2.31% | Facebook | 2.54% | Facebook | 2.43% |
| 8 | Facebook | 2.02% | MPEG Streaming | 1.98% | SSL | 1.74% |
| 9 | Flash Video | 1.34% | iTunes | 1.68% | MPEG Streaming | 1.66% |
| 10 | BBC iPlayer | 1.27% | SSL | 1.54% | iTunes | 1.53% |

**Picture 1.** Upstream traffic, downstream traffic and half in Europe.

(http://velocidad.es/etiqueta/p2p/)

We can see that the P2P programs (BitTorrent and eDonkey) accumulate approximately the 30 per cent of the traffic on Internet.

## Applications

P2P technology is extensive and can reach many areas, the transmission of files can be useful for a lot of operations.

We see a list of the most important applications developed with the help of P2P:

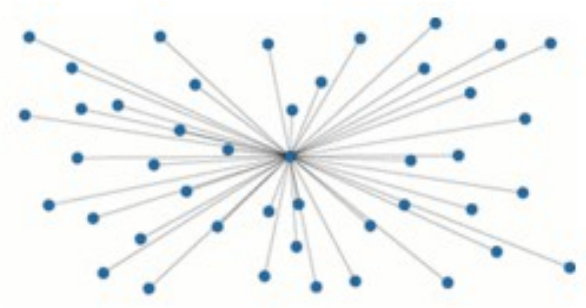
- Transferring large files. The most extended application of this type of network.
- Distributed file systems like CFS or Freenet.
- Transfer of Posts: between MTAs (mail transport agents)
- Routing Protocols (information on host connections).
- Multimedia: Watch video on demand. For example TVUPlayer and PPLive.
- Peer-to-peer content services, caches for improved performance such as Correli Cache.
- Spotify uses a peer-to-peer network along with streaming servers to stream music to its desktop music player.
- Distributed search engine, a search engine where there is no central server.
- VoIP (using application layer protocols such as SIP).
- Instant messaging and online chat.

- Internet telephony systems. Skype is also supported in part by a P2P system.
- Virtual currencies for transactions between parties. (Bitcoin)
- Research. Scientific calculations to process huge databases.
- Provide anonymity and freedom of expression (i2p, or MorphMix Tarzan.).
- Groups of news (Usenet).

## Classification of networks

P2P networks base their operation on person-to-person, but have certain ratings that help us be grouped into four classes:

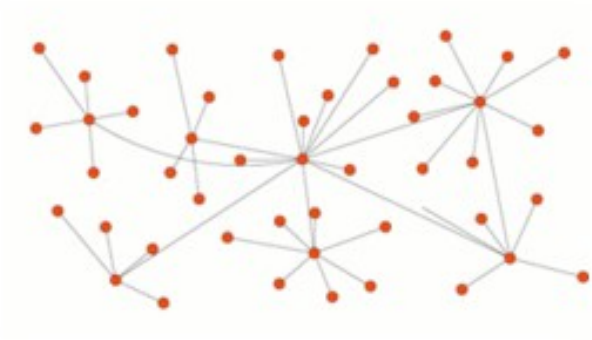
- Centralized: These networks depend on central server equipment that runs the operation, for example **Napster**.



**Picture 2. Centralized network**

(http://es.wikipedia.org/wiki/Peer-to-peer)

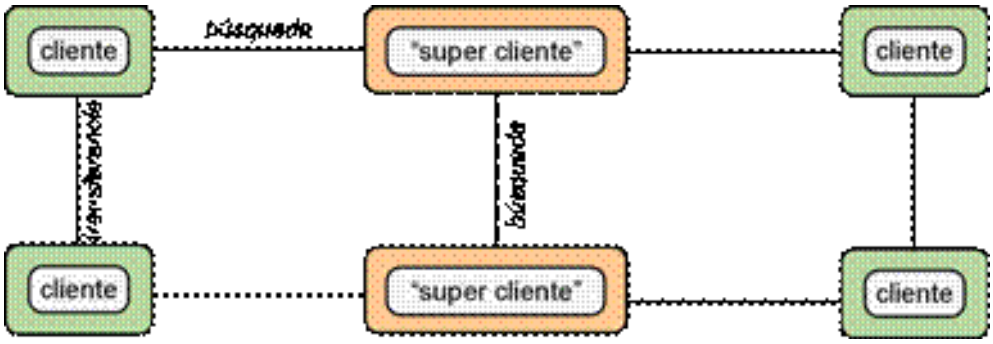
- Decentralized: Do not depend on specific equipment, and each node contains information that is shared, for example Freenet.



**Picture 3. Decentralized network.**

(http://es.wikipedia.org/wiki/Peer-to-peer)

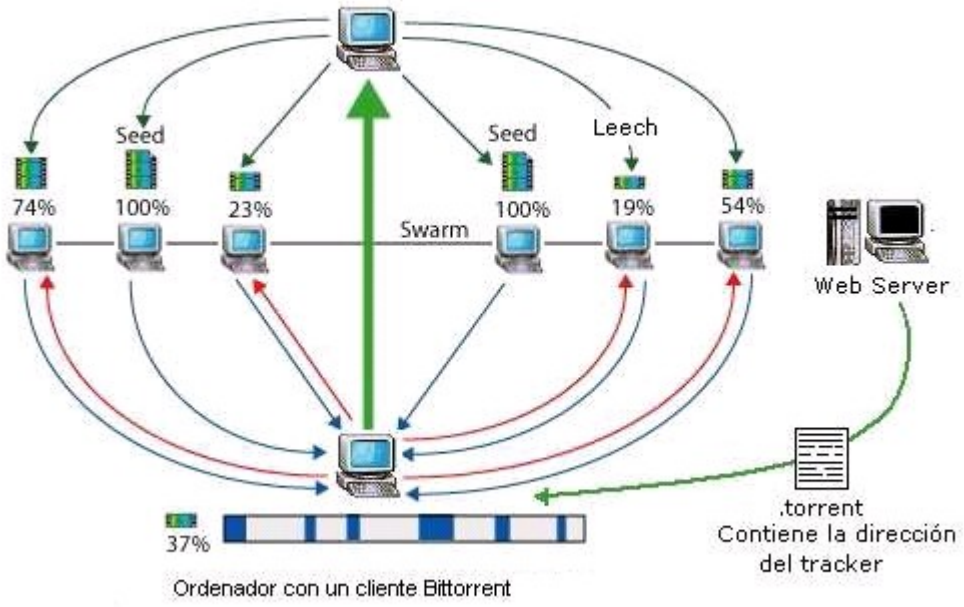
• Semi - Centralized: A mixture of the two types of networks previously seen.




**Picture 4. Semi-centralized network.**

(http://www.ub.edu/geocrit/sn/sn-170-54.htm)

• Structured: categories exist that allow control nodes on the network structure, for example **BitTorrent**.




**Picture 5. BitTorrent network.**

(http://es.wikipedia.org/wiki/Archivo:Red_bittorrent.jpg)

• Unstructured: Connections and the overall structure is arbitrary, for example **Gnutella**.

Also, networks can be classified according to the way they conduct searches and manage the files, we can distinguish two main groups:

- Global Networks: In this type of networks is only a network of servers that provide service to all users, is normally operated by a company or organization, for examples of these networks are Freenet, Kazaa, Gnutella and OpenFT.

- Private Networks: In such networks there are no mains, only there are many protocol-based networks that may or may not be interconnected. For example Bittorrent and DirectConect.

## Types of P2P

- **Bittorrent**

SEMI-CENTRALIZED, STRUCTURED and LOCAL (each tracker operates independently of other, each with its rules and content) network.
This is the program that most data moves across the Internet, because it gets to move files from one computer to another are large.

The shared files on the network are handled through a file called "torrent" containing metadata to share files (tracker, the user, hashes, file list, comments, etc.), the torrent is distributed through a common network (such as HTTP or other P2P network), when a user wants a file normally goes to the page tracker holder (although there are sites that list many torrent trackers) and download the torrent file (less than 500kb). When the user opens the file with a torrent client, it communicates to the tracker, which tells of active users, and the client makes direct connections to each of the nodes and download parts of the file. As the client is downloaded parts, is able to serve them to other nodes.

Its main customers are Bittorrent (official client), Azureus (Java) and uTorrent (C++ for Windows).

- **eDonkey**

SEMI-CENTRALIZED (There isn't an only central server.), STRUCTURED and GLOBAL network.

This is the number 1 networks of P2P. eMule is the most popular client of eDonkey, about 80% of users use this program.

**eDonkey is based on central servers that keep connections with users and that are responsible for keep an index of files**, customers **are connected to each other via ad-**hoc **connections and multiple connections can exist**.

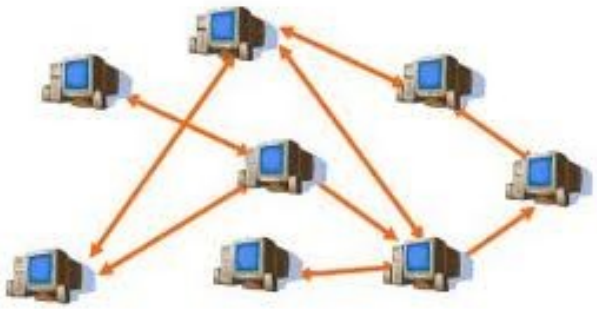
- **Gnutella**

DESCENTRALIZED, DESESTRUCTURED and GLOBAL network.

Today is the third most used P2P network in the world.

Gnutella is a completely decentralized network, where different methods are used to achieve higher communication speeds and free network overloads.

The network is based on the interconnections of the nodes to exchange information, downloads are done by direct connection active or passive. Searches were initially structured by the flooding mechanism (the node asks its adjacent nodes and their adjacent these) but then implemented distributed index tables.



**Picture 6. Gnutella network.**

(http://digitalrightscorp.com/joomla/index.php?

option=com_content&view=article&id=78&Itemid=472)

Ultra peer: thus called nodes capable of containing new nodes and store search indexes.

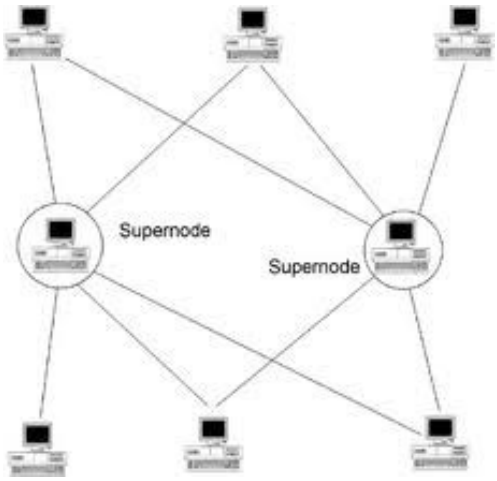
- **FastTrack (Kazaa)**

Is one of the most used P2P networks, is the official Kazaa client.

SEMI-CENTRALIZED (There isn't an only central server.), STRUCTURED and GLOBAL network.

The operation is in hybrid P2P as it is done by special nodes (called super nodes) which are responsible for maintaining temporary indexes of the files on the network.

It has not a central server.




**Picture 7. FastTrack network.**

(http://flylib.com/books/en/3.446.1.11/1/)

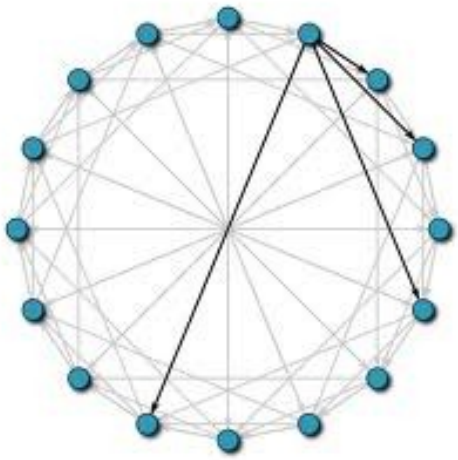
- **Chord**

DESCENTRALIZED and STRUCTURED network.

Chord is a simple and scalable protocol for distributed search in P2P networks linking key with nodes.

It is a network of 2 ^ n nodes (active or absent). Given a set of keys in the network, the Chord protocol is responsible for assigning keys to nodes existing assets and keep those assignments dynamically. The assignment of identifiers and keys to nodes is performed using a hash function consistent.

Chord is designed to be highly scalable, so that changes in the size of the system do not significantly affect performance. In particular, if N is the number of nodes of the network, its cost is proportional to log (N).



**Picture 8. Chord network.**

(http://www.romascanu.net/Blog/the-chord-protocol-a-neat-solution-for-distributed-peer-to-peer-networks/)

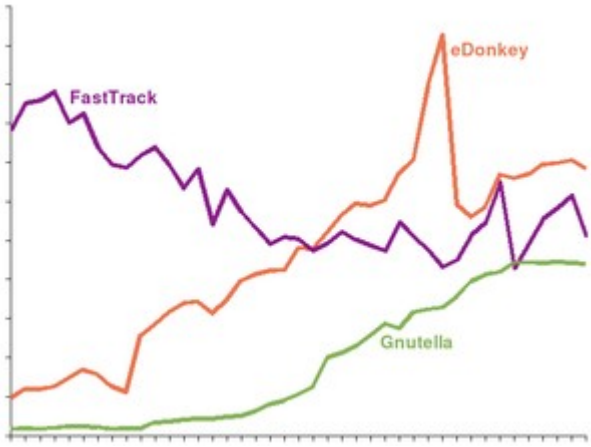
- **Direct Connect**

  CENTRALIZED, STRUCTURED and PRIVATE network.

  This network is based on servers called hubs (principal server in network), which are responsible for maintaining a list of clients connected to the network. Each hub is managed privately by the owner of the system in which it resides, and has control over the rules of it, the normally function as community hub, with a main chat and regulated.

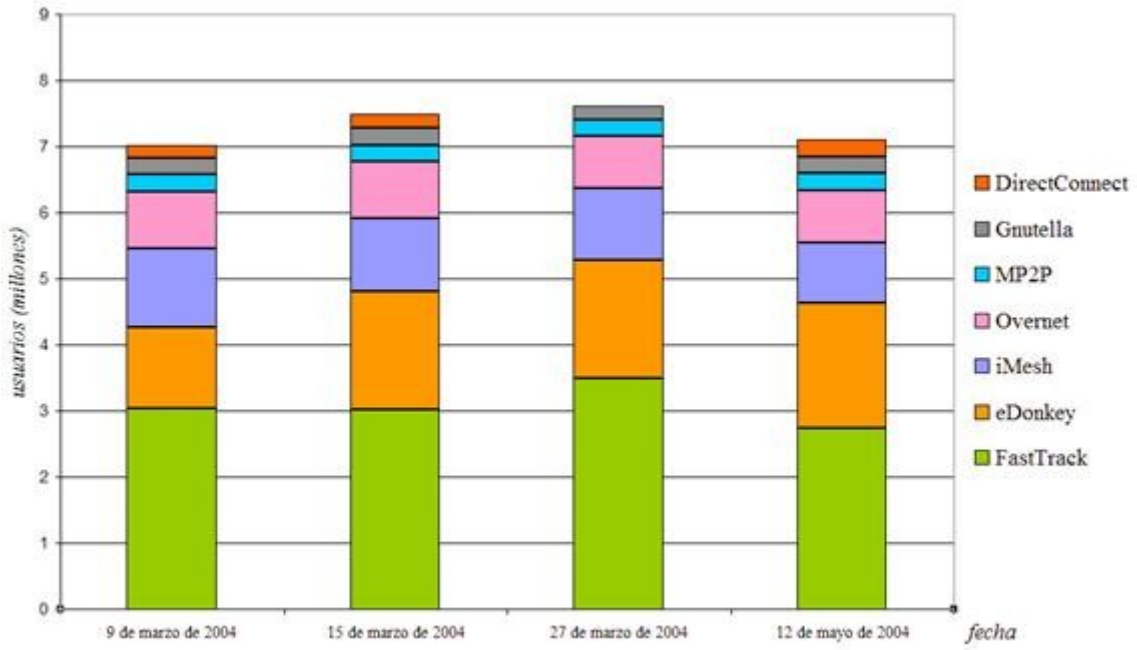
## Comparison of different systems

Note the number of users over time:




**Picture 9. Number of users of network FastTrack, eDonkey and Gnutella, from January 2003 to May 2006.**

(http://es.wikipedia.org/wiki/FastTrack)




**Picture 10. New user in the principal p2p programs.**

(http://www.ub.edu/geocrit/sn/sn-170-54.htm)

## Problems in P2P (Attacks)

P2P networks promote free file sharing and services, so it is more vulnerable to certain types of attacks:

- **Poisoning.**

  Is to distribute information and/or files that do not correspond to the description given (false files).

- **Pollution.**

  Insert pieces or packages within packages malignant legitimate, sometimes only affect performance and containing other malicious programs.

- **Saturation.**

  Traffic caused by P2P networks can be excessive for the systems, causing choking the network, but also can be orchestrated attacks to produce a denial of service.

- **Identification.**

  Some P2P networks allow users to identify, which have taken enough IP agencies to locate users.

- **Scamming.**

  Many people take advantage of the ignorance of the user and will sell special services, which are nothing more than pay for a service that is free.

## Chat program

Now let's focus on the application to develop the project, which is a chat, in which connect two computers using a TCP connexion to send messages from one to the other.

To connect and the sending and receiving of messages for simplicity we have used the Socket class that has already defined these methods in its API (Application Programming Interface).

For a better visualization we added a graphical interface in which appear the conversation developed earlier and from which we can continue to send other messages.

For all this we will use Java as a programming language, in addition to develop the graphical interface use the NetBeans program, we will develop the components to display.

## Client operations

Let's see the principals operations that the client has:

- **CONNECT**

How connect with the other peer?

The server is waiting to clients.

"St" is the IP of the other node (will be introduced this number in the graphical interface) and 1111 is a random port that we chosen:

*s = new Socket(st,1111);*

When I connect, I can get the channel where send and receive message from the other peer and I save in "c_in" and "C_out".

*c_in = s.getInputStream();*

*c_out = s.getOutputStream();*

- **SEND MESSAGE**

When I want to send a message, first I need calculation the length of this string and include this number before the string for that the other peer read first this number and later receive only this number of bytes and show on the screen.

```
length = text.length();

//add the length at the string in the message

m[0]=(char)((length/100)+48);

m[1]=(char)(((length%100)/10)+48);

m[2]=(char)((length%10)+48);

//Join length + message

String message = String.valueOf(m)+text;
```

When I have the length and message together, it sends this to other peer.

```
n=message.getBytes();


c_out.write(n);
```

- **RECEIVE MESSAGE**

When I am waiting to receive a message, I receive first the length of the string and I know the number of bytes that need to read.

We read the 3 first numbers:

```
while(received<3){

        received+=c_in.read(a,received,3-received);

}

//Size of String

size=(new Integer(new String(a))).intValue();

received=0;

while(received<size) {

        received+=c_in.read(b,received,size-received);

}
```

Now, I have the string in the variable “received”.

## Server operations

The server only have to wait to client wants connect and later, when the client is already connected receive message and will answer with the same message changed with to upper letters.

```
ServidorTCP serv=new ServidorTCP();

serv.arrancar();//waiting to client wants connect

//Always available

while(true){

    n=serv.recibir();

    n=n.toUpperCase();

    serv.enviar(n);

}
```

The server has the same methods than the client but it works different because the client write text and the server only receive and answer.

The server is waiting in the same port than the client wants to connect.

```
servidor=new ServerSocket(1111);
```

```
System.out.println("waiting");

//Accept only one connection

canal=servidor.accept();
```

## Graphical interface

Our application graph is a "front-end." Because it is an application that provides an interface to another.

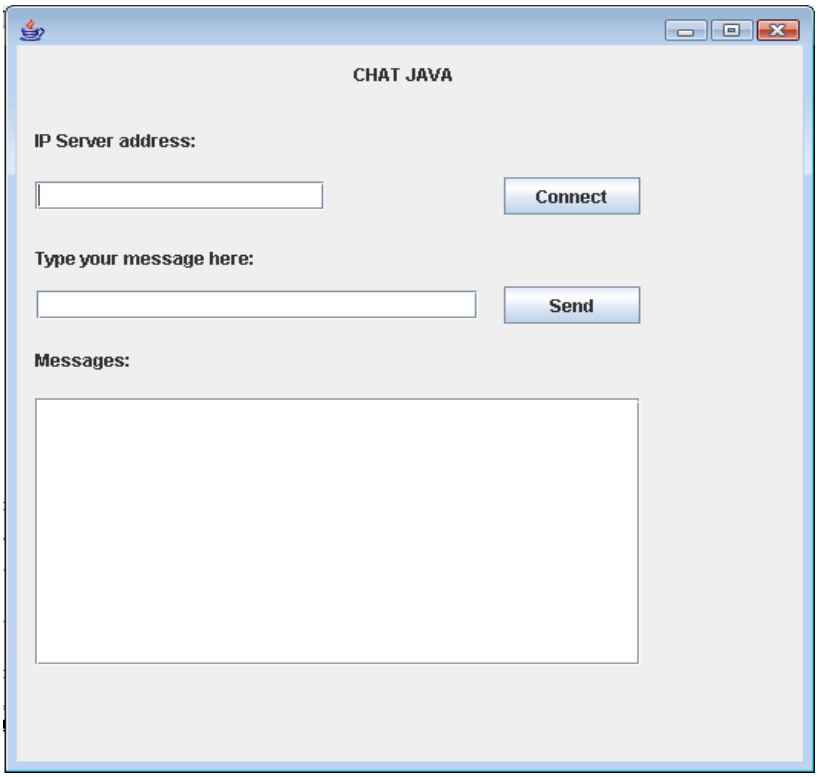
Now let's see an example with the graphical interface of the application running, to see the steps that have to happen to function properly.

Example:

First of all, in particular the graphical application that is designed to develop all the required functionality.

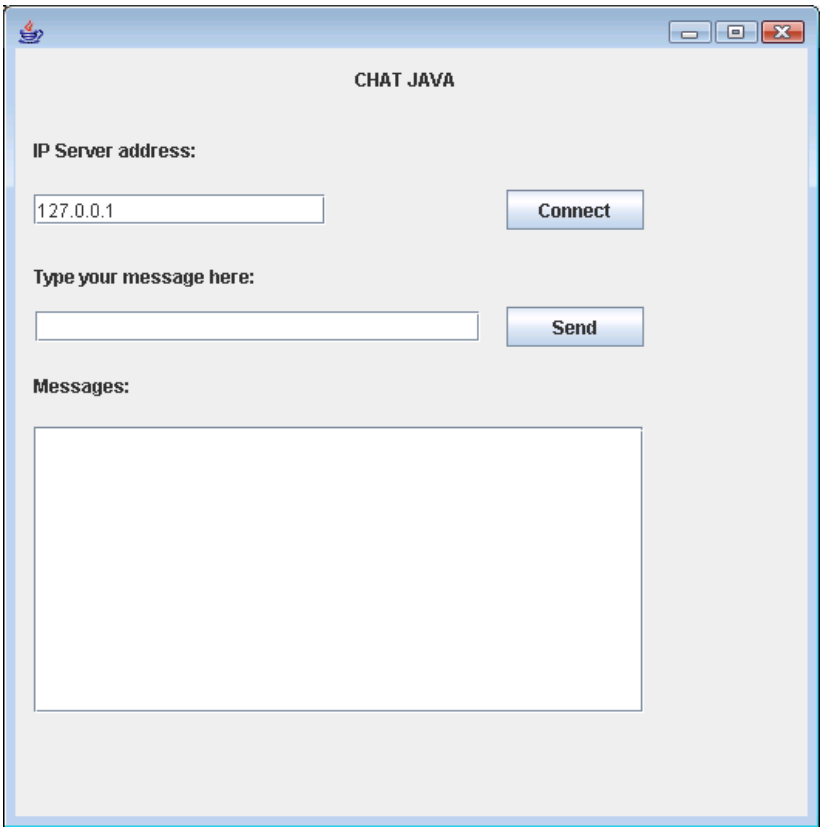
Contains several holes to introduce information like the IP address, the text to send to the server when we are connected, several buttons to send information (push to connect and send your text), plus a larger hole in which each entry the entire conversation with the server.

First execute the server (which will remain waiting for the client to connect) and then to the customer. When we execute the client appear:

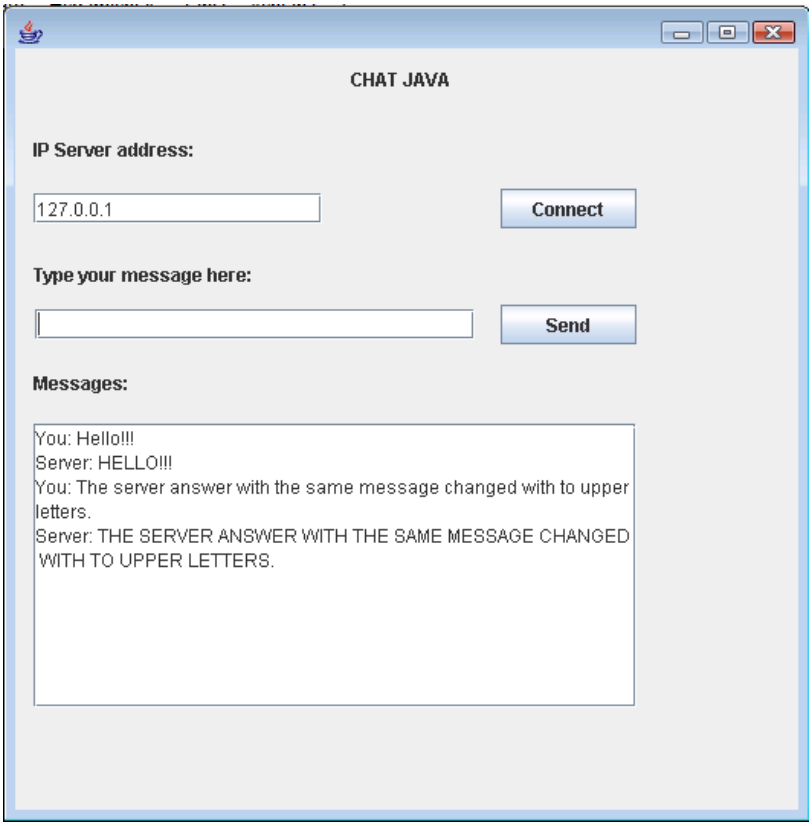



To work correctly you have to enter the destination IP address, for our example we choose the IP address 127.0.0.1 because I have only one computer for testing.

Press the button and connect the server and the client will connect if we don’t have any problem occurs.




After that we can already begin to send text to the server, which will have to be included under the text "type your message here" and then press the "Send" button.




As we move to the server sending messages anger responding with the same content but modified in uppercase as shown in the image above.

## Multiplies clients

Now modify the server so that it can respond to multiple clients.

To do this we need to make multiple threads launched server will come, according to new requests from clients, we make use of thread class java api.

With this kind execute an object of the class when we receive a new connection request to the server and may well serve all simultaneously, taking for each different object.

For verifying system operation need several computers requesting the connection to the server and sending text simultaneously.

## How extend the program?

To it more similar to a peer-2-peer chat this we would need to include more complexity to the application:

We would have to ask a central server for the file you are looking for specifically.

This would answer with the IP address of any other node that contains the information sought (could also provide a list of those who have the file) or not give us anything if none has that file.

From there we'd head to one of these nodes (or several of them) to ask for information such as the IP address we would make a request to connect and would ask that file.

It would also be necessary to enable the transfer of information that was not only text and allow the transfer of more data (ability to attach files).

## RESUMEN EN ESPAÑOL

## Universidad Carlos III de Madrid

Miguel Jiménez García

Ingeniería de Telecomunicaciones

Cotutor en la UC3M: IVAN LOZANO GUADALAJARA ilguadal@it.uc3m.es

Coordinador académico: DANIEL CUESTA

Universidad destino: BME Budapest University of Technology and Economics

Cotutor: CZIRKOS Zoltán

Fecha de lectura: 2013

Calificación obtenida: 3 (NOTABLE)

## INTRODUCCIÓN

En mi proyecto el objetivo era desarrollar una aplicación en un lenguaje cualquiera que tuviese las características de una aplicación P2P.

Inicialmente tuve que hacer un estudio sobre las diferentes características y utilidades de este tipo de aplicaciones, obteniendo así una visión general de ellas, y viendo hacia donde habían evolucionado en los últimos años.

La aplicación elegida fue un chat simple en el que varios ordenadores compartiesen información. La información tenía que ser procesada por el receptor y devuelta al emisor, para así comprobar que la recepción era correcta.
Para el desarrollo de la aplicación utilice el lenguaje de programación Java, ya que por parte de mi tutor no importaba el lenguaje a utilizar.

## TECNOLOGÍA P2P

P2P significa peer to peer, sin una red o software, sino que se define como una estructura de red o una forma de organización lógica.
El uso principal o lo que más se utiliza para compartir archivos entre varios usuarios, ya que tiene muchas más utilidades.

Es una red de computadoras en las que los clientes y servidores cambian todo el

tiempo, los nodos (Peers = cada uno los sistemas conectados a la red) se comportan contrarios a las redes en condiciones de igualdad, cliente-servidor. Es decir, actúan simultáneamente como clientes y servidores a otros nodos de la red.
Redes P2P permiten el intercambio directo de información, en cualquier formato, entre los ordenadores conectados en red.

Esto significa que el P2P no define un protocolo o reglas específicas para su uso, el P2P sólo indica cómo se deben hacer las conexiones y la organización de los nodos.

En las transferencias P2P puede ser muy rápido, ya que tienen conexiones con todo el equipo capaz de proporcionar el servicio. Varias partes se descargan de muchas fuentes diferentes al mismo tiempo (descarga de varias partes).
El protocolo es más complejo que una estructura de cliente-servidor.

## Aplicaciones

Vemos las aplicaciones más importantes que se desarrollan con la ayuda de P2P:

• La transferencia de archivos de gran tamaño. La aplicación más extendida de este tipo de red.
• Transferencia de Mensajes: entre los MTA (Agente de transporte de correo)
• Protocolos de enrutamiento (información sobre las conexiones de host).
• Multimedia: Ver video bajo demanda. Por ejemplo TVUPlayer y PPLive.

• Los servicios de contenido Peer-to-peer, cachés para mejorar el rendimiento como Correli caché.

• Spotify utiliza una red peer-to-peer, junto con servidores de streaming para transmitir música a su reproductor de música de escritorio.

• VoIP (utilizando protocolos de capa de aplicación, tales como SIP).

• Mensajería instantánea y chat en línea.

• Los sistemas de telefonía por Internet. Skype también está apoyado en parte por un sistema P2P.

• Investigación. Cálculos científicos para procesar grandes bases de datos.

## CLASIFICACIÓN DE LAS REDES

Redes P2P basan su funcionamiento en persona a persona, pero tienen ciertas calificaciones que nos ayudan a ser agrupadas en cuatro categorías:

• Centralizado: Estas redes dependen de equipos servidor central que ejecuta la operación, por ejemplo Napster.

• Descentralizado: No dependen de equipos específicos, y cada nodo contiene la información que se comparte, por ejemplo Freenet.

• Semi-centralizada: Una mezcla de los dos tipos de redes visto anteriormente.

• Estructurado: existen categorías que permiten los nodos de control en la estructura de la red, por ejemplo BitTorrent.

• No estructurados: Conexiones y la estructura general es arbitraria, por ejemplo, Gnutella.

Además, las redes pueden ser clasificados de acuerdo a la forma en que llevan a cabo búsquedas y gestionar los archivos, se pueden distinguir dos grandes grupos:

• Las redes globales: En este tipo de redes es sólo una red de servidores que dan servicio a todos los usuarios, es normalmente por una empresa u organización, para ejemplos de estas redes son Freenet, Kazaa, Gnutella y OpenFT.

• Redes Privadas: En este tipo de redes no hay red, sino que hay muchas redes basadas en protocolos que pueden o no pueden estar interconectadas.
Por ejemplo Bittorrent y DirectConect.

## Tipos de redes P2P.

Vemos las diferentes versiones mas importantes que hay definidas como redes P2P, no incluyo las características de cada uno, solamente la clasificación a la que se encuentra del apartado anterior.

- **Bittorrent**

Red SEMI-CENTRALIZADA, ESTRUCTURADA y LOCAL.

- **eDonkey**

Red SEMI-CENTRALIZADA, ESTRUCTURADA y GLOBAL.

- **Gnutella**

Red DESCENTRALIZADA, DESESTRUCTURADA y GLOBAL.

- **FastTrack (Kazaa)**

Es una de las redes P2P mas usadas.

Red SEMI-CENTRALIZADA, ESTRUCTURADA y GLOBAL.

- **Chord**

Red DESCENTRALIZADA y ESTRUCTURADA.

- **Direct Connect**

Red CENTRALIZADA, ESTRUCTURADA y PRIVADA.

## Programa chat

Ahora vamos a centrarnos en la aplicación para desarrollar el proyecto, que es un chat, en el que conectar dos ordenadores mediante una conexión TCP para enviar los mensajes de una a la otra.

Para conectar, envío y recepción de mensajes por simplicidad, hemos utilizado la clase Socket que ya ha definido estos métodos en su API (Interfaz de Programación de Aplicaciones).

Para una mejor visualización añadimos una interfaz gráfica en la que aparece la conversación desarrollada anteriormente y con la que podemos continuar enviando otros mensajes.

Por todo esto vamos a utilizar Java como lenguaje de programación, además de desarrollar la interfaz gráfica con el programa NetBeans, con la que vamos a desarrollar los componentes para mostrar.

Operaciones de clientes

Veamos los directores de operaciones que el cliente tiene:

• Conectar

El servidor está esperando a los clientes.

Cuando me conecto, consigo el canal en el envío y recibo mensajes de los otros nodos.

• Enviar mensaje

Cuando quiero enviar un mensaje, primero tengo que calcular la longitud de esta cadena e incluir este número antes de la cadena para que el otro nodo sepa primero este número y después recibe sólo el número de bytes necesarios y

mostrarlos en pantalla.

• RECIBIR MENSAJES

Cuando estoy a la espera de recibir un mensaje, recibo primero la longitud de la cadena (tres primeros bytes) y sé el número de bytes que hay que leer.

**Operaciones del servidor**

El servidor sólo tiene que esperar al cliente que quiere conectar y más tarde, cuando el cliente ya está conectado recibir el mensaje y responderá con el mismo mensaje cambiado a letras mayúsculas.

El servidor tiene los mismos métodos que el cliente pero funciona diferente, porque el texto de escritura cliente y el servidor sólo reciben y contestan.
El servidor está esperando en el mismo puerto que el cliente desea conectarse.

## Interfaz gráfica

Vamos a diseñar una Interfaz grafica para que podamos utilizar nuestro programa de una manera sencilla e intuitiva.

En primer lugar, la aplicación gráfica que está diseñado para desarrollar toda la funcionalidad requerida.

Contiene varios huecos para introducir información como la dirección IP, el texto a enviar al servidor cuando estamos conectados, varios botones para enviar información (pulsar para conectarse y enviar su texto), además de un agujero más grande para escribir el texto a mandar al servidor.

Para que funcione correctamente tiene que introducir la dirección IP de destino, para nuestro ejemplo elegimos la dirección IP 127.0.0.1 ya que tengo un solo ordenador.

Presione el botón y conectar el servidor y el cliente se conectará si no tenemos ningún problema.

Después de esto ya podemos empezar a enviar mensajes de texto al servidor.

A medida que avanzamos en el envío de mensajes el servidor irá respondiendo con el mismo contenido, pero modificado a mayúsculas.

## Muchos clientes

Modificamos el servidor para que pueda responder a múltiples clientes.

Para ello tenemos que hacer varios subprocesos lanzaron por el servidor, hacemos uso de clase Thread del API de Java.

Con este tipo podemos ejecutar un objeto de la clase cuando recibimos una nueva solicitud de conexión al servidor y puede servir a todos al mismo tiempo, ya que cada cliente tiene un objeto diferente.

Para verificar el funcionamiento del sistema necesita varios ordenadores solicita la conexión al servidor y enviar mensajes de texto al mismo tiempo.

## Agradecimientos

En este punto, me gustaría dar las gracias a todas las personas que han contribuido directa o indirectamente en mis años de estudio en esta carrera.

A todos los profesores que me han impartido clase en la universidad y que me han preparado para ahora dar el salto al mundo laboral.

Por último no me quiero olvidar de mis compañeros de universidad, familia y amigos, que me han estado ayudando durante todos estos años de carrera y que gracias a ellos todo es siempre mucho más fácil.

Gracias a todos.